Attorney Docket
No. 201521.0190

"EXP[redacted]S MAIL" MAILING LABEL NO.[redacted]74258708US

TITLE OF THE INVENTION

System And Method For Communicating, Monitoring
And Configuring A Device Operatively Connected To A Network

BACKGROUND OF THE INVENTION

The present invention comprises a system, as well as a method for monitoring a network and for configuring a device, such as a printer, which is operatively linked to the network or a computer. The system is also for establishing communication between the network and the device in a format understandable by a human operator of the network. The present system and method is capable of communicating with virtually any type of operatively linked device utilizing any available device configured protocol and can generate a user interface in any supported human understandable language. The system can also automatically update itself with respect to newly developed devices or changes to device protocols by downloading any needed data from a central location using an existing communication network, such as the Internet.

Devices, such as printers are typically described by various configuration variables using a variety of different system protocols and languages including SNMP, PJL, HTTP, etc. machine languages which are not easily understood by humans. However, in each language there is a code or series of codes or characters which describe or identify a commonly performed task, instruction or characteristic. For example, all printers have a characteristic which in English is "Paper Tray Source". The same characteristic in a PJL variable would be "OKIPAPERFEED" and in a SNMP variable would be "1.3.6.1.4.1.2001.1.1.1.1.1.40". The present invention communicates with connected devices utilizing the system protocol or language of the device and "translates" information/data from the connected device into a selected human understandable format for presentation to a human operator.

The present invention reduces development and maintenance time and cost by building generic applications that are modular, data driven and utilize dynamic graphical user interface (GUI) generation instead of integral programs with hard-coded mapping of data to GUI. Using this approach the same software program can operate with any type of device and can present a user interface in any human understandable language. If a device of a previously unknown type is discovered by a system or network, the system or network can dynamically and automatically update itself to permit communication with the newly discovered device or to

generate a user interface in a new language. Many network management applications have "knowledge" about device variables coded into the management program. With the present invention description of devices are stored externally so that a generic program can be used making different network protocols, system and human languages and device types and can obtain needed device configurations and variables from the external source as needed.

BRIEF SUMMARY OF THE INVENTION

Briefly stated, the present invention comprises a method of monitoring a network for communicating with a device operably connected to the network. The network employs a data engine for interfacing with a data agent connected to the device. The method comprises detecting the presence of the device through the data agent interfacing the data engine; communicating with a first external repository, the repository for storing device information about the device; selecting information corresponding to the device from information stored at the first repository; and transferring the selected information from the first repository to the data engine for use in communicating with the device, whereby the network is dynamically upgraded as devices are operatively linked thereto.

BRIEF DESCRIPTION OF THE SEVERAL VIEW OF THE DRAWING

The foregoing summary, as well as the following detailed description of preferred embodiments of the invention will be better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there are shown in the drawings, embodiments which are presently preferred. It should be understood, however, that the present invention is not limited to the precise arrangements and instrumentality shown. In the drawings:

Fig. 1 is a functional schematic block diagram of a first embodiment of a system 10 in accordance with the present invention;

Figs. 2a and 2b illustrate examples of portions of the software function of the system of Fig. 1; and

Figs. 3-6 are software flow diagrams and sequence diagram illustrating two examples of the operation of the system of Fig. 1.

DETAILED DESCRIPTION OF THE INVENTION

The system 10 in the present embodiment is specifically directed to peripheral devices, particularly printers, which are connected or otherwise operatively linked to a network or computer. It should be understood that the present invention is equally applicable with respect to devices other than printers, which may be connected or otherwise operatively linked to a network or computer. The system 10 operates under the assumption that every device may be "described" by a set of property variables that are NAME-VALUE pairs. To find out a device configuration, the system can "GET" variable and to change a device configuration, the system can "SET" the values of the device variables.

As illustrated in Fig. 1, the system 10 includes a network manager or data engine 12 which is employed for controlling the processing of the various functions of the system 10. In the present embodiment, the data engine 12 is a software module which receives data and/or variables in a predetermined system language from other network components (hereinafter described) and translates or dynamically assembles the received data/variables into a selected, human understandable presentation language or format, such as, GUI, HTML, etc.. The data engine 12 thereafter utilizes one or more user interface software modules to dynamically generate a user interface for the data/variables to facilitate user access to the system. Examples of such user interface software modules include an operating system such as Windows 14, a web browser 16 such as Internet Explorer and an E-mail server 18 and a COM interface 17. Other such user interface software modules will be apparent to those of ordinary skill in the art. Each user interface, module communicates with the data engine 12 using a specific address. The user interface modules communicate with a user by dynamically generating screens, web pages, e-mail, etc.

The system 10 also includes a plurality of network agents or data agents 20 which are operatively connected to the data engine 12 to collect and to present data. Each data agent 20 is a software module that communicates with a device, in the present embodiment, printers 22a, 22b, 22c, which are operatively connected to the network. As previously

mentioned, any other suitable type of device could alternatively be operatively connected to a data agent 20, if desired. Each data agent 20 "discovers" and communicates with a connected device, such as one of the printers 22a, 22b, 22c, using the specific system protocol and/or language (such as SNMP, PJL, etc.) which is used by the particular device. For example, in connection with the present embodiment, a first data agent 20 communicates with a parallel port printer 22a, utilizing a first standard printer language such as, PJL another data agent 20 communicates with a first network printer 22b utilizing a second standard printer language such as SNMP, etc. The data engine 12 selects a particular data agent 20 that is associated with a selected device. By communicating with the attached device in the system protocol or language used by the device a data agent 20 can obtain from the device information concerning the identity of the device, as well as the language of the attached device and necessary configuration variables. The data agent 20 provides data in the form of NAME=VALUE arrays as well as additional flags like READ ONLY and a listing of allowed values from the device. The obtained data concerning the device is presented by the data agent 20 to the data engine 12. The communication between the data agent 20 and the connected device is preferably bi-directional to facilitate changing the value of configuration variables of the device as selected by a user or as needed.

The system 10 further includes a data dictionary 24 in the form of a local description cache, preferably in XML format. The data dictionary 24 stores information to facilitate translations between the various system protocol or languages used within the devices 22a, 22b, 22c and provides such translation information to the data engine 12 to assist and facilitate the data engine 12 in translating data from the various devices into the selected human understandable form. The data engine 12 thus uses the data dictionary 24 to obtain variable names in the language of a selected data agent 20. For some devices which communicate with the system 10, the data dictionary 24 initially contains in its memory sufficient information to facilitate the translation of the data by the data engine 12. In the case of a new or unknown device connected to the network or an updated device detected by a data agent 20, for which no data or insufficient date is stored within the data dictionary 24, it may be necessary to dynamically update the data dictionary 24 with the requisite information. The data dictionary 24 may be dynamically updated by accessing a central repository of information or data.

The central data repository, referred to in the present embodiment as data central 26 is preferably at a centralized location and is accessible by a variety of communications

systems or networks. If necessary, the data central 26 may obtain data which it does not already have from a second repository or source, such as from a vendor's website or the like. In the present embodiment, the data central 26 is accessible by way of the Internet. As shown in Fig. 1, the data central 26 may be accessed directly by the data dictionary 24. Alternatively, a request for the required information may be sent directly to the data central 26 from the data engine 12 and, may either be downloaded directly to the data dictionary 24 or may be supplied to the data dictionary 24 through the data engine12. The data central is contacted to obtain required information which is not available to the data engine 12 through the data agents 20 or the data dictionary 24. Preferably, data or information is downloaded from the data central 26 in XML format utilizing the HTTP protocol. Once the required translation data concerning the device has been stored in the data dictionary 24, the data engine 12 may access the data, as required for performing its translation functions with respect to the device.

Fig. 2a is a example of formatted data from the data dictionary 24 or data central 26 with variable descriptions in XML based language. The <VARIABLES> may contain multiple <VARIABLE> elements. Each <VARIABLE> element has a "KEY" attribute and multiple child <NAME> elements with translations. Fig. 2b is an example of formatted data from a data agent 20 formatted in XML. The root XML element is <VARIABLES> which may contain multiple <VARIABLE> elements with "NAME", "VALUE" or "OPTIONS" attributes and with <OPTION> child elements with various possible variable values. Both of the examples of Fig. 2a and Fig. 2b are provided as samples. Real syntax may vary depending upon the requirements of a particular application.

Figs. 3 and 4 are standard program flow diagrams using module: function descriptions in the oval boxes with the diamond shaped boxes being branching points. In Figs. 3 and 4, the module names are abbreviated as follows:

UI - User Interface Module

DD - Data Dictionary Module

DA - A Data Agent Module

DE - The Data Engine Module

DC - Data Central

Figs. 5 and 6 are sequence diagrams in standard Unified Modeling Language

(UML). In Figs. 5 and 6, the horizontal boxes across the top and the module names as used above. In Figs. 5 and 6 time is shown as progressing from the top of the diagrams to the bottom and the arrows represent messages passing among the modules. The vertical boxes or dashed vertical lines represent the time of the module activity.

Figs. 3 and 5, taken together are an example of the process of getting translated names and values of all variables. At step 302, the user interface requires values for all variables for each device at a given address using a selective language. At step 304, the data engines selects the particular data agent based upon the address for the device. At step 306, the data dictionary provides the identification names of variables based upon the data agent and address. If the required data is not in the data dictionary, the data central gets the data at block 308 and updates the data dictionary at block 310. At block 312, the data agent provide values from the required address for selected variable names. If the data agent can not provide the required data, the data engine tries another data agent at block 304 and the process repeats. At block 314, the data engine determines the device type from the values of the variables and at block 316, the data dictionary provides names of all of the variables that are relevant for the particular type of device and for the particular data agent. If the data dictionary does not have the list of variables, the names are obtained from the data central at block 318 and, at block 320, the data dictionary is updated. At block 322, the data agent gets the values of the variables for the list of all variables. At block 324, the data dictionary translates the values to the selected language. If insufficient information is available in the data dictionary to perform the translation, the information is obtained from data central at block 326 and at block 328, the data dictionary is updated with the newly obtained data. Finally, at block 330, the user interface module is presented with the translated names and values of all of the variables. The same sequence of steps can be followed by referring to the UML sequence diagram of Fig. 5.

Figs. 4 and 6 illustrate the steps involved in changing a value of a variable. At block 402, the user interface module selects a device address and variable key. At block 404, the data engine selects the agent based upon the selected address. If the agent is available, at block 406, the data dictionary provides the variable name for a given variable key and agent and at block 408 the data agent provides possible values for the variable. If there is no available data agent, at block 410 the data engine tries to determine the device type based upon the address and attempts to get possible values at block 412. If the data dictionary does not contain the possible values they are obtained from the data central at block 414 and the data

dictionary is updated at block 416. At block 418, the data dictionary translates the possible variable values to the selected language. If the data dictionary does not have the information necessary for the translation, at block 420 the information is obtained from data central and at block 422, the data dictionary is updated. At block 424, the user interface presents translated possible values of the variables and the user selects the new value. At block 426, the data engine again selects the agent and at block 428, the data dictionary translate the new variable value to the selected data agent language. If the data dictionary does not have sufficient information to complete the translation, at block 430, the translation information is obtained from data central and at block 432, the data dictionary is updated with the new information. At block 434, the data agent changes the variable value. The process describe above and as shown in connection with Fig. 4 can be followed by reference to the UML sequence diagram of Fig. 6.

From the foregoing, it can be seen that the present invention comprises a novel system and method for communicating, monitoring and configuring a device operatively connected to a network. It will be appreciated by those skilled in the art that changes could be made to the embodiments described above without departing from the broad inventive concept thereof. It is understood, therefore, that this invention is not limited to the particular embodiments disclosed, but it is intended to cover modifications within the spirit and scope of the present invention as defined by the appended claims.